Glowworm swarm optimisation algorithm for virtual machine placement in cloud computing

Alboaneen, Dabiah Ahmed; Tianfield, Huaglory; Zhang, Yan

Published in:
Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences

DOI:
10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0129

Publication date:
2017

Document Version
Author accepted manuscript

Glowworm Swarm Optimisation Algorithm for Virtual Machine Placement in Cloud Computing

Dabiah Ahmed Alboaneen*†, Huaglory Tianfield*, and Yan Zhang*
*School of Engineering and Built Environment, Glasgow Caledonian University, Glasgow, G4 0BA, United Kingdom
†College of Education, University of Dammam, Jubail, Kingdom of Saudi Arabia
Email: {dabiah.alboaneen; h.tianfield; yan.zhang}@gcu.ac.uk

***Abstract*—Virtual machine placement (VMP) is the assignment of virtual machines (VMs) to physical hosts (PHs). In this paper, we apply a glowworm swarm optimisation (GSO) algorithm to solve the VMP problem so that the energy consumption and the service level agreement (SLA) violation are minimised. Simulation results show that GSO based VMP algorithm outperforms many of the common VMP algorithms.**

***Index Terms*—cloud computing; resource management; energy efficiency; glowworm swarm optimisation (GSO); virtual machine placement.**

I. Introduction

The process of placing virtual machines (VMs) on physical hosts (PHs) is referred to as the virtual machine placement (VMP). There are two types of initial conditions for VMP problems: (1) fresh VM placement where a new VM is placed on PH, and (2) VM re-placement which is the optimisation of the existing placement of VMs. The main difference is that in VM re-placement, live VM migration is used to move a VM from one PH to another without noticeable service interruption [1].

The need for re-placing VMs is due to the change in the data centre (DC) environment, such as workload variations or hardware failures. Generally, applications located in VMs are usually associated with service level agreements (SLA). After a period of time, violations of SLA may occur due to factors such as high CPU utilisation or high memory usage of the PH. Hence, some VMs need to be migrated to avoid over-utilisation that causes VM performance degradation. On the other hand, some PHs may be switched off or turned to low-power modes to reduce the energy consumed by the underutilised PHs. Therefore, the optimal placement plays an important role in improving energy efficiency and resource utilisation in a cloud environment.

The VMP problem in cloud computing is a kind of a bin-packing problem and a non-deterministic polynomial-time hard (NP-hard) problem [1]. Generally, it is difficult to develop algorithms for producing optimal solutions within a short time. Metaheuristic techniques such as genetic algorithm (GA), ant colony optimisation (ACO) and particle swarm optimisation (PSO) can deal with these problems by providing near-optimal solutions within a reasonable time.

In this paper, we apply a glowworm swarm optimisation (GSO) to VMP. The proposed GSO based VMP (GSOVMP) algorithm takes into account the dynamic requirements of users in a cloud environment include live migration as an option in order to utilise resources more efficiently.

To evaluate the performance of GSOVMP, we undertake comparative simulation study of the proposed algorithm with a power aware best fit decreasing (PABFD) algorithm [2]. Simulation results show that GSOVMP outperforms PABFD in terms of energy, SLA violations and the number of VM migrations.

The main contribution of this paper lies at exploring GSO algorithm to solve energy and SLA aware VMP problem. CloudSim [3] is used to evaluate the performance of the GSOVMP algorithm.

The remainder of this paper is organised as follows. Section II presents a review on VMP in a cloud environment. The VMP problem is formulated in Section III. The GSO algorithm is described and discussed in Section IV. Simulations setup and the results are examined in detail in sections V and VI. Finally, in Section VII, the paper is concluded and future work is mentioned.

II. Literature Review

The most common VMP algorithm may be PABFD algorithm [2]. The idea of PABFD is to sort all VMs in decreasing order of current CPU utilisations and then to allocate the new VM to the PH that provides the smallest increase in the power consumption caused by allocation.

Metaheuristics approaches have been studied for solving the VMP problem. Kaaouache *et al.* proposed a GA algorithm for VMP [4]. Wu *et al.* used simulated annealing (SA) [5], Ali and Lee used biogeography-based optimisation (BBO) [6] and Wang *et al.* used PSO [7] for energy-efficient VMP.

However, the existing work did not take use of the live VMs migration to re-optimise the VMP and was mostly focused on the VMP with fresh initial conditions. Single-objective problem VMP of minimising the energy consumption is solved by using ACO [8] and [9], PSO [10], SA [11] and GA [12], respectively.

Multi-objective VMP problem is also studied. Xiong *et al.* [10] addressed the efficient usage of multi-dimensional resources to minimise the energy consumption via PSO. In [13], [14] and [15] a multi-objective VMP problem is solved by using GA to minimise the number of active PHs and the communication traffic and to balance multi-dimensional resources used simultaneously within the DC. Moreover, GA

TABLE I: Power consumption by two types of PHs at different load level in Watts

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

was used for VMP to reduce the power consumption by minimising the number of active PHs and maximising the efficiency of resource usage [16]. In addition, ACO was also used to minimise the power consumption and resource wastage [17], [18] and [19]. The ACO was used by Liu *et al.* for VM re-placement to reduce the energy consumption and load balancing for resources [20]. In [21], multi-objective VMP problem is solved by using BBO to decrease power consumption, resource wastage, server unevenness, inter-VM traffic, storage traffic and migration time simultaneously. In [22], multi-objective VMP problem is solved by using memetic algorithm to minimise the energy consumption, network traffic and to maximise the economical revenue.

However, the trade-off between energy and performance is an important issue and needs to be addressed in VMP problem. In [23], SA algorithm achieves a balance between the energy consumption and the number of VMs migration. Gao and Tang used the PSO algorithm for VMP to improve the resource utilisation of the PH and minimise the number of VM migrations [24]. In [25], PSO based VMP algorithm achieves trade-off between energy and performance. In [26], the authors used ACO to find a near-optimal solution for the re-placement of VMs to improve the resource utilisation of PHs and to reduce the energy consumption.

Our work aims at minimising energy consumption and number of VM migrations. Moreover, minimising SLA violations is included in our VMP objectives.

III. VMP Problem Formulation

For convenience and clarity, notations are listed in Table II.

A. Assumptions

We assume a single DC with heterogeneous PHs is provisioning multiple VMs. Compared with other physical resources such as memory and bandwidth, the main fraction of the energy is consumed by the CPU rather than by the memory and bandwidth [27]. In [28] it is found that there exists a strong relationship between CPU utilisation and total power consumption by a PH. The power consumption by a PH increases linearly with the increase of CPU utilisation from the power consumption at the idle status up to the power consumption at the PH's is full utilisation. Hence, in this paper we assume that the power consumption of a PH is linear to its resource utilisation (i.e., CPU utilisation).

B. Power Consumption Model

The real power consumption data provided in SPECpower benchmark is used for calculating the power consumption. Two different PHs are used in our simulations: HP ProLiant ML110 G4 and HP ProLiant ML110 G5. The power consumption by PHs at different load levels is presented in Table I.

TABLE II: Indices and notations

j	Index for PHs.
M	Total number of PHs.
E_j	Energy consumption of PH j.
$Pcpu_j$	CPU capacity of PH j.
$Pmem_j$	Memory capacity of PH j.
$Pnet_j$	Network bandwidth capacity of PH j.
i	Index for VMs.
N	Total number of VMs.
$Vcpu_i$	CPU demand of VM i.
$Vmem_i$	Memory demand of VM i.
$Vnet_i$	Network bandwidth demand of VM i.
$\ell_j(t)$	Luciferin value of PH j.
$p_{jn}(t)$	Probability of VM in PH j to select PH n.
$x_j(t)$	CPU utilisation of PH j.
$x_j(t+1)$	New CPU utilisation of PH j.
$\gamma_d^j(t)$	Local radial range j.
γ_s	Max sensor range of the $\gamma_d^j(t)$.
$N_j(t)$	Neighbours set of PH j.
$\|N_j(t)\|$	Actual number of neighbours.
n_t	Number of desired neighbours.
$\|\|x\|\|$	Euclidean norm of x.
β	Change rate of the neighbourhood range.
s	Step size of moving.
γ	Luciferin enhancement coefficient.
p	Luciferin decay coefficient $(0 < p < 1)$.

C. Objective Function

The objective function of the GSOVMP algorithm focuses on minimising the energy consumption E. The objective function f is written as follows:

$$f = \sum_{j=1}^{M} E_j \tag{1}$$

where j is index of PH and E_j is the energy consumption of PH j.

D. Constraints

- Constraint 1: A VM must be assigned to a PH, i.e.,

$$\forall i, \sum_{j=1}^{M} Vp_{ij} = 1 \tag{2}$$

where i is index of VM, j is index of PH and Vp_{ij} is a binary value representing whether VM i is assigned to PH j.

- Constraint 2: The total resources of a VM cannot exceed the capacity of the PH resources, i.e.,

$$\forall j, \sum_{i=1}^{N} Vcpu_i * Vp_{ij} \leq Pcpu_j \tag{3}$$

$$\forall j, \sum_{i=1}^{N} Vmem_i * Vp_{ij} \leq Pmem_j \tag{4}$$

$$\forall j, \sum_{i=1}^{N} Vnet_i * Vp_{ij} \leq Pnet_j \tag{5}$$

where $Vcpu_i$, $Vmem_i$ and $Vnet_i$ are CPU, memory and network bandwidth demands of VM i, respectively. $Pcpu_j$, $Pmem_j$ and $Pnet_j$ are CPU, memory and network bandwidth capacities of PH j , respectively.

Algorithm 1 GSO

Initialise parameters β , p, s, n_t
$\forall j$, set $\ell_j(0) = \ell_0$
$\forall j$, set $\gamma_d^j(0) = \gamma_0$
while $termination\_condition\_not\_meet$ **do**
  for j = 1 **to** m **do**
    $\ell_j(t+1) = (1-p)l_j(t) + \gamma f(x_j(t+1))$
    $N_j(t) = \{n : ||x_n(t) - x_j(t)|| \leq \gamma_d^j(t); \ell_j \leq \ell_n(t)\}$
    for each $n \in N_j(t)$ **do**
      $p_{jn}(t) = \frac{\ell_n(t) - \ell_j(t)}{\sum_{k \in N_j(t)} \ell_k(t) - \ell_j(t)}$
    end for
    $x_j(t+1) = x_j(t) + s\left(\frac{x_n(t) - x_j(t)}{||x_n(t) - x_j(t)||}\right)$
    $\gamma_d^j(t+1) = \min\{\gamma_s, \max\{0, \gamma_d^j(t) + \beta(n_t - |N_j(t)|)\}\}$
    $t \leftarrow t+1$
  end for
end while

IV. Glowworm Swarm Optimisation-Based Virtual Machine Placement (GSOVMP) Algorithm

GSO algorithm is a swarm intelligence algorithm developed by Krishnan and Ghose. GSO is a population-based algorithm. As control is not centralised at a single point, thus it is more scalable [29].

GSO is based on the behaviour of glowworms. A glowworm that produces more light (high luciferin) means that it is closer to an actual position and has a high objective function value. Each glowworm selects a neighbour that has a higher luciferin value than its own, according to a probabilistic mechanism, and moves towards it. These movements are based only on local information. Therefore, it enables the glowworms to divide into subgroups leading to the detection of multiple optima of the given objective function.

GSO algorithm starts by positioning glowworms randomly in the workspace and all the glowworms contain an equal quantity of luciferin. A GSO algorithm comprises four phases, i.e., glowworm initialisation, luciferin updating, glowworm moving and local radial range updating. The pseudocode of GSO algorithm is presented in Algorithm 1.

GSO algorithm will be applied to search for the placement of VMs that minimises the energy and SLA violations. Then we have the four phases of the process of GSOVMP as follows.

A. Glowworm Initialisation Phase

The initial population consists of a number of glowworms, which are considered as candidate solutions to the problem. A glowworm is a vector of elements, and an element represents a VM; the value of an element is the ID of the PH to which the VM is assigned. Here, the initial population is generated based on the least loaded PH.

Each PH j has a location $x_j(t)$ that is defined by CPU utilisation, a luciferin value $\ell_j(t)$ as calculated in Eq. (6), which represents the available CPU utilisation, a local radial range $\gamma_d^j(t)$, max sensor range γ_s , the size of moving step s, the number of desired neighbours n_t, the luciferin decay coefficient (0<p<1), the luciferin enhancement coefficient γ and the change rate of the neighbourhood range β.

$$\ell_j(t) = Pmax_j - P_j \tag{6}$$

where $Pmax_j$ is the maximum power of the PH j according to the power model and P_j is the current power of the PH j based on CPU utilisation.

B. Luciferin Update Phase

Initially, each PH has its own luciferin value. Each PH j converts the objective function value $f(x_j(t+1))$ at its current location $x_j(t)$ to a luciferin value $\ell_j(t+1)$ by using the formula below.

$$\ell_j(t+1) = (1-p)l_j(t) + \gamma f(x_j(t+1)) \tag{7}$$

where $\ell_j(t)$ is the luciferin value of PH j at time t. p is the luciferin decay coefficient ($0 < p < 1$), γ is the luciferin enhancement coefficient, and $f(x_j(t+1))$ represents the value of the objective function of PH j's location at time t.

Therefore, the luciferin values of PHs are changed according to the objective function values. Whenever a VM is placed to a PH, luciferin value of this PH is updated. A higher luciferin value means a better result as the goal is to minimise the energy consumption. Also, the luciferin value is decreased along the time to simulate the decay.

C. Movement Phase

In the movement phase, a VM in PH j chooses to move toward one of its neighbours n that has a higher luciferin value (more available CPU utilisation) and within the local radial range γ_d. The movement consists of three further steps.

- Step 1. Find Neighbours: The set of neighbour in the local radial range can de written as below.

$$N_j(t) = \{n : ||x_n(t) - x_j(t)|| \leq \gamma_d^j(t); \ell_j \leq \ell_n(t)\} \tag{8}$$

where $N_j(t)$ is the neighbour set, n is the index of PH close to PH j, $x_n(t)$ and $x_j(t)$ are locations of PH n and PH j, respectively, $\ell_j(t)$ and $\ell_n(t)$ are luciferin values for PH j and PH n, respectively. $||x||$ is the Euclidean norm of x, and $\gamma_d^j(t)$ represents the local radial range.

- Step 2. Calculate Probabilities: For each PH j, the probability to figure out the movement direction toward the neighbour with a higher luciferin value is calculated by using the formula below.

$$p_{jn}(t) = \frac{\ell_n(t) - \ell_j(t)}{\sum_{k \in N_j(t)} \ell_k(t) - \ell_j(t)} \tag{9}$$

where $p_{jn}(t)$ is the probability of VM in PH j to move to PH n.

The VM located in PH j selects a PH n from the neighbour set that has the highest probability over others in the neighbour set.

- Step 3. Movement: At the end of the movement phase, the location of the VM is changed based on the location of

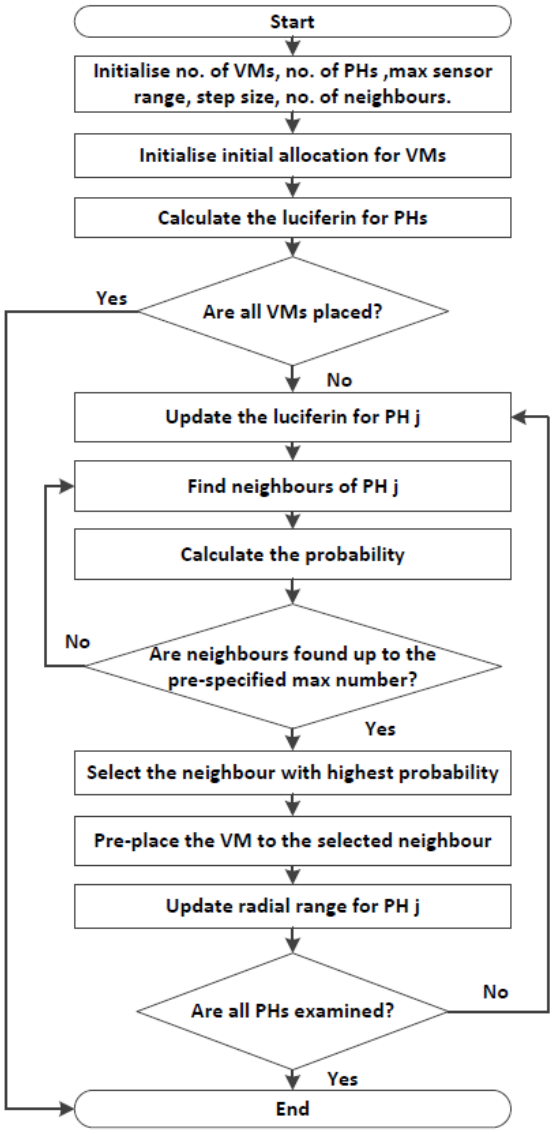


Fig. 1: The GSOVMP flowchart

the selected PH. The new location of the VM is calculated using the formula below.

$$x_j(t+1) = x_j(t) + s\left(\frac{x_n(t) - x_j(t)}{||x_n(t) - x_j(t)||}\right) \quad (10)$$

where $x_j(t+1)$ and $x_j(t)$ are the new and current locations for VM j, respectively, s is the step size of moving.

D. Local Radial Range Update Phase

The local radial range γ_d^j is updated using Eq. (11) in order to formulate the neighbour set in the next VMP for adding more flexibility to the PH.

$$\gamma_d^j(t+1) = \min\{\gamma_s, \max\{0, \gamma_d^j(t) + \beta(n_t - |N_j(t)|)\}\} \quad (11)$$

where β is a coefficient and n_t is the number of desired neighbours. $|N_j(t)|$ is the actual number of neighbours.

The flowchart and pseudocode of the GSOVMP algorithm are shown in Fig 1 and Algorithm 2, respectively. GSO

Algorithm 2 GSOVMP

```
1: Input: PhList, VM, set of parameters
2: Output: Allocation of VMs
3: Initialise parameters β , p, s, n_t, γ_d^j
4: g = newGlowworm(vm, PhList)
5: PhId = luciferin.getPh()
6: currentluc=g.calculateLuc(PhId) by Eq.(1)
7: while i < N do
8:   for j = 1 to M do
9:     newluc=g.calculateNewLuc(PhId) by Eq.(7)
10:    neighbours = g.getNeighbours(PhId,neighbourSize) by Eq.(8)
11:    while j < neighbours.size() do
12:      neighbourId = neighbours.get(j)
13:      neighbourluc=g.calculateLuc(neighbourId)
14:      if neighbourluc = 0 then
15:        currentluc = neighbourluc
16:        neighbourPhId = neighbours.get(j)
17:        j= neighbours.size()
18:      end if
19:      luc = currentluc − neighbourluc
20:      if luc > luciferin then
21:        if neighbourPhId < γ_d^j then
22:          luciferin = luc
23:          currentluc = neighbourluc
24:          neighbourPhId = neighbours.get(j)
25:        end if
26:      end if
27:    end while
28:    allocatedPh = PhList.get(neighbourPhId)
29:    Update radial range by Eq.(11)
30:  end for
31: end while
```

parameters are initialised (lines 3-4). When a user requests a VM, a PH is initialised based on the least loaded PH through $getPh()$ method (line 5) and each PH gets a location as per its utilisation by calculating the luciferin through the $calculateLuc(PhId)$ method (line 6). For each VM i that needs to be migrated (line 7), the algorithm will search for a suitable PH j (line 8): The luciferin of PH will be updated (line 9). The neighbour set will be calculated (line 10), it contains PHs which have a higher luciferin value than the original one and are within the local radial range. The size of the neighbour set is predefined by the user. The luciferin of a neighbour PH will be calculated (line 13). If a neighbour PH is not suitable for hosting the migrated VM, then this neighbour PH will not be taken into consideration (lines 14-18). The luciferin is calculated as the available utilisation difference between the original PH j and its other neighbour PHs (line 19). If any of the neighbouring PHs is less utilised than the original PH and is within the local radial range, then this PH will be the selected neighbour of PH (lines 20-24). The selected neighbour will be the allocated PH to place the VM (line 26).

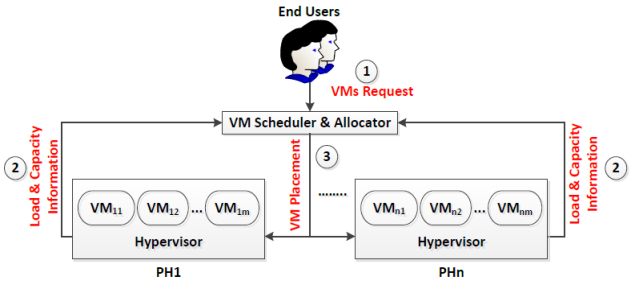


Fig. 2: The system architecture

After that, the radial range which defines the neighbour set will be updated (line 27). Finally, the algorithm will be terminated when suitable PHs for placing all the migrated VMs are found (line 29).

V. Simulation Environment

A. System Context of Resource Management

Suppose a large-scale DC consists of 800 heterogeneous PHs and four VM types are modelled based on Amazon EC2 instance types. A PH is characterised by the amount of CPU, memory and network bandwidth. The system architecture is shown in Fig 2. Users submit requests for the provisioning of N VMs, as marked by information (1), which are allocated to PHs by executing a VMP algorithm. The length of each request is specified in millions of instructions (MI). Initially, VMs are allocated according to the requested characteristics assuming 100% CPU utilisation. Afterwards, the proposed VMP algorithm optimises VMP to reduce the power consumption, SLA violations and VM migrations of the DC. One or more VMs can run in one PH, and each PH runs a hypervisor. Monitoring of the CPU utilisation of PHs and detection of overloaded and underloaded PHs are fed back, as marked by information (2). The VM scheduler and allocator (VSA) collects the feedback information from the PHs to maintain an overall view of the system resource utilisation. Based on the information of user requests and the feedback from PHs, the VSA initiates the VMP algorithm to optimise the allocation of VMs and hypervisors perform actual migration and placement of VMs and changes in power modes of PHs.

To sum up, the resource management system works as follows.

(i) Initialise a placement for each VM based on least loaded PHs.

(ii) Monitor CPU utilisation of PHs by applying a PH load classification algorithm. In our simulation, local regression (LR) is used to classify PHs into different states based on the following rules:

- If the current CPU utilisation exceeds the PH capacity, the PH is regarded as overloaded.
- If the current CPU utilisation is less than a threshold of the total CPU utilisation, the PH is treated as underloaded.

(iii) Determine which VMs should be migrated by applying a VM selection algorithm. In our simulation, we use minimum migration time (MMT) to select VMs to migrate.

(iv) Re-place the migrated VMs in new PHs by applying the proposed GSOVMP algorithm.

(v) Switch off inactive (idle) PHs if the CPU utilisation is 0%.

B. Simulation Setup

CloudSim toolkit 3.0.3 simulator [3] was used to evaluate the proposed algorithm. CloudSim is widely used to simulate cloud system components such as DCs and VMs. It supports policies for VM placement and selection, power models for DC resources, and provides different types of workloads.

The results are based on real workload which is provided as part of the CoMon project, a monitoring infrastructure for PlanetLab [30]. Five workload traces collected by Beloglazov and Buyya in March 2011 [2] are used. During the simulations, each VM is randomly assigned a workload trace from one of the VMs from the corresponding day. The number of VMs in each day is shown in Table III.

TABLE III: The number of VMs in each workload day

Workload No.	1	2	3	4	5
Date	03/03	06/03	09/03	22/03	25/03
No. of VMs	1052	898	1061	1516	1078

As mentioned in Krishnanand and Ghose [29], the choice of GSO parameters has some influence on the performance of the algorithm. In terms of the total number of peaks captured, Krishnanand and Ghose suggested the parameter selection as tabulated in Table IV. Thus, only n_t and γ_s need to be selected.

The implementation of the proposed algorithm in CloudSim is carried out using the *optimizeAllocation* method to optimise the current VMP. This method takes VMs list as a parameter and returns a map of the best placement found. Specifically, the GSO algorithm will run in the *findHostForVm* method for both the *getNewVmPlacementFromUnderUtilizedHost* and *getNewVmPlacement* methods.

VI. Results And Discussion

To evaluate GSOVMP algorithm, we compare it with the PABFD algorithm in [2]. In addition, we consider the LR algorithm to monitor CPU utilisation and decide whether a VM needs to be migrated or not. The main idea of LR is to set upper and lower utilisation thresholds and keep the total CPU utilisation of a PH between them. When the utilisation of a PH exceeds the limit, VMs are re-placed. Furthermore, to select the migrated VMs, we choose the MMT algorithm. The reason for choosing LR and MMT is that these algorithms outperform other existing algorithms in CloudSim as demonstrated in [2].

TABLE IV: The GSO algorithm parameters selection

Parameter	p	r	β	nt	s	l_0
Value	0.4	0.6	0.08	5	0.03	0.05

A. Energy Consumption

This metric represents the total energy consumed by the physical DC resources. As illustrated in Fig 3a, the energy consumption for GSOVMP in all workloads is less than that with PABFD by $21.30\% - 27.18\%$. This is because that GSO algorithm is able to search the solution space more efficiently. GSO algorithm searches for the PHs to place VMs within local range and defined number of neighbours PHs and thus can find the solutions with a smaller number of PHs.

B. Number of VM Migrations

This metric represents the total number of migrated VMs. The fewest number of VM migrations is the best for decreasing performance degradation. The total number of migrations is based on the required CPU utilisation of VMs. In Fig 3b, the proposed algorithm GSOVMP has fewer VM migrations than PABFD in all workloads. This is due to the fact that PABFD leads to an increase in the number of overloaded and underloaded PHs which result in more VM migrations. In addition, it can be observed that increasing the number VM migrations caused an increase in energy consumption because VM migration consumes some of the CPU on the migrating PH.

C. SLA Violations

We consider two types of SLA violations in the infrastructure as a service (IaaS) layer: SLA violation per active host (SLAVH) and SLA violation due to migration (SLAVM). The SLAV is calculated as below.

$$SLAV = SLAVH * SLAVM \tag{12}$$

SLAVH is the ratio of $SLAV$ time of PH and the active time of this PH and is calculated as below.

$$SLAVH = \frac{1}{M} \sum\nolimits_{j=1}^{M} \frac{T_{s_j}}{T_{a_j}} \tag{13}$$

where M is the number of PHs; T_{s_j} is the total time that PH j has experienced the utilisation of 100% leading to an SLA violation. T_{a_j} is the total duration of PH j being in the active state.

SLAVM is the ratio of the performance degradation of VM caused by migrations and the overall CPU capacity requested by this VM. It is calculated as below.

$$SLAVM = \frac{1}{N} \sum\nolimits_{i=1}^{N} \frac{C_{d_i}}{C_{r_i}} \tag{14}$$

where N is the number of VMs, C_{d_i} is the estimate of the performance degradation of VM i caused by migrations and C_{r_i} is the total CPU capacity requested by VM i during its lifetime. We estimated C_{d_i} as 10% of CPU utilisation in MIPS during all migrations of VM i. It can be seen from Fig 3c that GSOVMP has a fewer SLA violations than PABFD in all workloads and the reason of this is that in the proposed algorithm, the number of VM migrations is fewer than with PABFD and the GSOVMP algorithm avoids PHs being 100% utilised.

D. ESV

As our goal is the trade-off between energy and SLA violation, it is important to consider a combined metric, e.g., a product of energy consumption and SLA violation as below.

$$ESV = Energy\_consumption * SLAV \tag{15}$$

In general, the ESV in GSOVMP is better than that in PABFD, as seen in Fig 3d.

VII. Conclusion and Future Work

We have applied GSO algorithm to solve the VMP problem and to find a near-optimal solution. We have proposed GSOVMP as a new VMP algorithm to improve energy efficiency and reduce SLA violations in cloud computing. The simulation results show that GSOVMP outperforms PABFD in terms of energy, SLA violation, the combination of energy and SLA violation and the number of VM migrations. Apparently it is interesting in the future work to evaluate the GSOVMP algorithm by comparing it with other metaheuristic algorithms and extending to more objective functions.

References

[1] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 10th IFIP/IEEE Int. Symp. on*, pp. 119–128, IEEE, 2007.

[2] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency Computation Practice and Experience*, vol. 22, no. 6, pp. 1397–1420, 2011.

[3] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[4] M. A. Kaaouache and S. Bouamama, "Solving bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud," *Procedia Computer Science*, vol. 60, pp. 1061–1069, 2015.

[5] Y. Wu, M. Tang, and W. Fraser, "A simulated annealing algorithm for energy efficient virtual machine placement," *IEEE Intl. Conf. on Sys., Man, and Cybernetics*, pp. 1245–1250, 2012.

[6] H. M. Ali and D. C. Lee, "A biogeography-based optimization algorithm for energy efficient virtual machine placement," *IEEE Symp. on Swarm Intelligence*, pp. 1–6, 2014.

[7] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," *Intl. Conf. on Parallel and Dist. Sys.*, pp. 102–109, 2013.

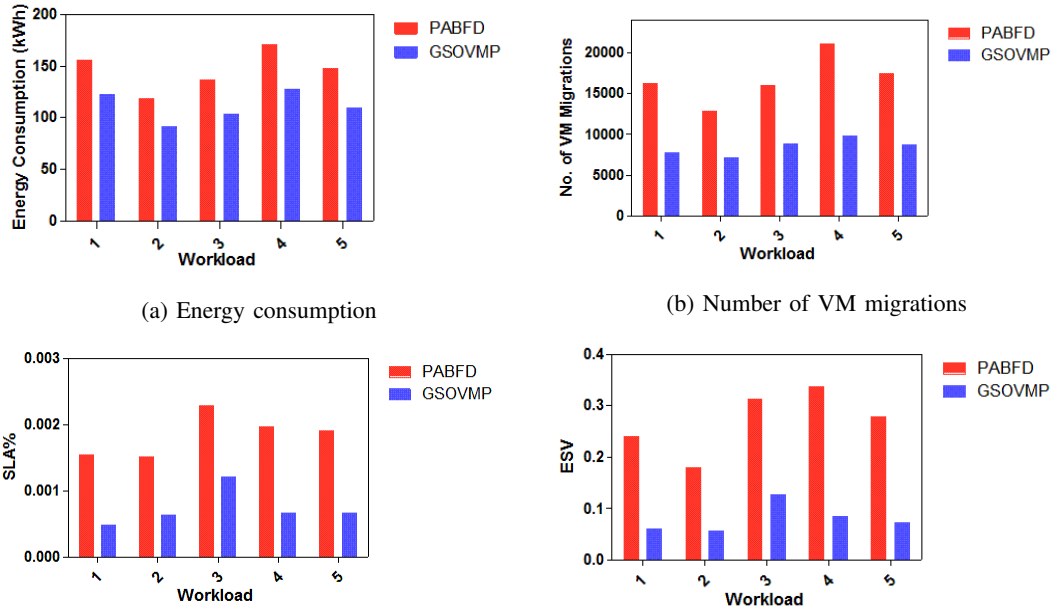
[8] E. Feller, L. Rilling, and C. Morin, "Energy-Aware Ant Colony Based Workload Placement in Clouds," *Grid Computing, 12th IEEE/ACM Intl. Conf. on*, pp. 26–33, 2011.

[9] X.-F. Liu, Z.-H. Zhan, K.-J. Du, and W.-N. Chen, "Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach," in *Proc. of the conf. on Genetic and evolutionary computation*, pp. 41–48, ACM, 2014.

[10] A.-p. Xiong and C.-x. Xu, "Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center," *Mathematical Problems in Eng.*, vol. 2014, pp. 1–8, 2014.

[11] N. Khalilzad, H. R. Faragardi, and T. Nolte, "Towards energy-aware placement of real-time virtual machines in a cloud data center," in *High Performance Computing and Communications, IEEE 7th Int. Symp. on CSS, IEEE 12th Int. Conf. on ICESS, IEEE 17th Int. Conf. on*, pp. 1657–1662, IEEE, 2015.

[12] N. K. Sharma and G. Reddy, "Novel energy efficient virtual machine allocation at data center using genetic algorithm," in *Signal Processing, Communication and Networking, 3rd Int. Conf. on*, pp. 1–6, IEEE, 2015.



(a) Energy consumption

(b) Number of VM migrations

(c) SLA violation

(d) Product of energy consumption and SLAV

Fig. 3: Comparisons of simulations. (a) energy consumption, (b) number of VM migrations, (c) SLA violation metric and (d) ESV metric.

[13] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications, IEEE/ACM Int. Conf. on & Int. Conf. on Cyber, Physical and Social Computing*, pp. 179–188, IEEE, 2010.

[14] C. Liu, C. Shen, S. Li, and S. Wang, "A new evolutionary multi-objective algorithm to virtual machine placement in virtualized data center," in *Software Eng. and Service Science, 5th IEEE Int. Conf. on*, pp. 272–275, IEEE, 2014.

[15] S. Jamali and S. Malektaji, "Improving grouping genetic algorithm for virtual machine placement in cloud data centers," *4th Intl. Conf. on Computer and Knowledge Eng.*, pp. 328–333, 2014.

[16] C. T. Joseph, K. Chandrasekaran, and R. Cyriac, "Improving the efficiency of genetic algorithm approach to virtual machine allocation," in *Computer and Communication Tech., Int. Conf. on*, pp. 111–116, IEEE, 2014.

[17] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "Virtual machine consolidation in cloud data centers using aco metaheuristic," in *Euro-Par Parallel Processing*, pp. 306–317, Springer, 2014.

[18] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and Sys. Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[19] M. a. Tawfeek, a. B. El-Sisi, a. E. Keshk, and F. a. Torkey, "Virtual machine placement based on ant colony optimization for minimizing resource wastage," *Communications in Computer and Information Science*, vol. 488, pp. 153–164, 2014.

[20] Z. Liu, Y. Xiang, and X. Qu, "Towards optimal cpu frequency and different workload for multi-objective vm allocation," in *Consumer Communications and Networking Conf., 12th Annual IEEE*, pp. 367–372, IEEE, 2015.

[21] Q. Zheng, R. Li, X. Li, and J. Wu, "A Multi-objective Biogeography-Based Optimization for Virtual Machine Placement," *15th IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Computing*, pp. 687–696, 2015.

[22] F. L. Pires and B. Baran, "Multi-objective Virtual Machine Placement with Service Level Agreement: A Memetic Algorithm Approach," *IEEE/ACM 6th Intl. Conf. on Utility and Cloud Computing*, pp. 203–210, 2013.

[23] A. Marotta and S. Avallone, "A simulated annealing based approach for power efficient virtual machines consolidation," in *Cloud Computing, IEEE 8th Int. Conf. on*, pp. 445–452, IEEE, 2015.

[24] J. Gao and G. Tang, "Virtual Machine Placement Strategy Research," *Intl. Conf. on Cyber-Enabled Dist. Computing and Knowledge Discovery*, no. 2, pp. 294–297, 2013.

[25] S. E. Dashti and A. M. Rahmani, "Dynamic VMs placement for energy efficiency by PSO in cloud computing," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 0, no. 0, pp. 1–16, 2015.

[26] F. Farahnakian, A. Ashraf, and T. Pahikkala, "Using Ant Colony System to Consolidate VMs for Green Cloud Computing," *IEEE Trans. on Services Computing*, vol. 8, no. 2, pp. 187–198, 2015.

[27] L. Minas and B. Ellison, *Energy efficiency for information technology: How to reduce power consumption in servers and data centers*. Intel Press, 2009.

[28] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, pp. 13–23, ACM, 2007.

[29] K. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications," *Multiagent and Grid Systems*, vol. 2, no. 3, pp. 209–222, 2006.

[30] K. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planetlab," *ACM SIGOPS Operating Sys. Review*, vol. 40, no. 1, pp. 65–74, 2006.